ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA DEL SOFTWARE

Análisis de algoritmos de inteligencia artificial para videojuegos

Analysis of artificial intelligence algorithms for videogames

Realizado por
David Rico Zambrana
Tutorizado por
Lorenzo Mandow Andaluz
Departamento
Dpto. Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, Noviembre 2016

Fecha defensa:
El Secretario del Tribunal

Resumen

Este documento contiene el Trabajo de Fin de Grado del alumno David Rico Zambrana, estudiante del Grado en Ingeniería del Software, en la Universidad de Málaga. Este trabajo se ha realizado bajo la tutorización de Lorenzo Mandow Andaluz, profesor del Departamento de Lenguajes y Ciencias de la Computación.
El trabajo se titula **Análisis de algoritmos de inteligencia artificial para videojuegos**, y consiste en el desarrollo de una aplicación usando Unity3D en la que se muestren de forma didáctica conceptos de inteligencia artificial aplicada a los videojuegos, como son el pathfinding o búsqueda de caminos y los steering behaviors o algoritmos de movimiento.
La aplicación tiene dos secciones diferenciadas. En una, se muestra al usuario un escenario de juego en el que se pueden seleccionar coordenadas de inicio y destino, y la aplicación muestra y compara los caminos que proporcionan los distintos algoritmos de pathfinding implementados. La otra sección de la aplicación consiste en un enfrentamiento en tiempo real entre un jugador humano y un jugador controlado por una inteligencia artificial. En esta sección se pueden controlar diversos parámetros de los steering behaviors que controlan al jugador IA.

Palabras clave: Búsqueda de caminos, algoritmos de movimiento, máquinas de estado, videojuegos, inteligencia artificial, Unity3D.

Abstract

This document contais the final dissertation ot the degree student David Rico Zambrana for the studies Grado en Ingeniería del Software, of Universidad de Málaga. This dissertation has been realised under the supervision of Lorenzo Mandow Andaluz, from the Departamento de Lenguajes y Ciencias de la Computación.
The title of this project is **Analysis of artificial intelligence algorithms for videogames**, and it consists in the development of a desktop application using Unity3D which shows in a didactic manner some concepts of artificial intelligence applied to videogames, as pathfinding or steering behaviors.
The application has two separate sections. In the first one, the user is presented with a game scenery in which start and goal coordinates can be chosen, and then the different paths calculated by the various pathfinding algorithms are shown to the user. The other section of the application consists in a real time confrontation between a human player and an artificial intelligence controlled player. In this section, the user can control an array of parameters which control the steering behaviors of the AI player.

Keywords: Pathfinding, Steering Behaviors, state machine, videogames, artificial intelligence, Unity3D.

Índice general

1 - Introducción

Este documento representa el trabajo de fin de grado del alumno David Rico Zambrana para los estudios de Grado en Ingeniería del Software de la Universidad de Málaga. El trabajo, bajo el título de "Análisis de algoritmos de inteligencia artificial para videojuegos", pretende mostrar de forma didáctica la aplicación de diversas técnicas de inteligencia artificial para juegos, así como comparar su efectividad y el realismo de sus resultados. Para ello, se ha desarrollado una aplicación en Unity3D en la que se ponen en práctica distintos conceptos, principalmente pathfinding y steering behaviors. Se trata de un juego de acción en tiempo real, en el que dos personajes se enfrentan en un escenario delimitado, representado con una perspectiva cenital. Los personajes pueden moverse en cualquier dirección, y al mismo tiempo pueden disparar en cualquier dirección (no necesariamente en la misma en que se mueven).

1.1. Motivación

El presente trabajo está motivado principalmente por un interés en el mundo del desarrollo profesional de videojuegos. El sector de los videojuegos es un sector cultural que cada vez tiene más importancia, y como industria mueve cifras equiparables o mayores a las del cine. Además, es una salida profesional del Grado en Ingeniería del Software que, sin embargo, no parece tener mucha consideración en los planes de estudios actuales. Es cierto que existen asignaturas que tratan aspectos específicos de los videojuegos, y que gran parte del conocimiento general es tan aplicable a los videojuegos como al desarrollo web, pero no deja de haber un vacío en cuanto a las herramientas usadas actualmente en la industria del videojuego, como pueden ser Unity3D o Unreal Engine, y también en cuanto a técnicas usadas comúnmente en el desarrollo.

1.2. Objetivos

Con este trabajo se pretenden conseguir múltiples objetivos:

- Desarrollar una aplicación en Unity3D, demostrando conocer una de las herramientas más usadas en el desarrollo de videojuegos profesional.
- Visualizar de forma didáctica los resultados de distintos algoritmos de búsqueda de caminos.

- Visualizar de forma didáctica el comportamiento de un agente controlado por steering behaviors, así como el efecto de variar diferentes parámetros de éstos.

1.3. Organización de la memoria

En este primer capítulo hemos introducido el trabajo realizado. A continuación, en los siguientes capítulos hablaremos de varios conceptos necesarios para el desarrollo de este trabajo (capítulo 2: antecedentes), luego se explicará el diseño y la arquitectura de la aplicación desarrollada (capítulo 3: diseño de la aplicación), un breve manual de usuario de la aplicación donde veremos qué cosas se pueden experimentar en cada sección (capítulo 4: uso de la aplicación), y por último las conclusiones que se extraen del presente trabajo así como áreas en las que sería interesante profundizar (capítulo 5: conclusiones y trabajo futuro). Un breve apéndice da unas pinceladas acerca de por dónde se puede extender este trabajo, y cerramos con las referencias bibliográficas y los agradecimientos.

2 - Antecedentes

En este capítulo se van a exponer diferentes conceptos relacionados con el desarrollo de este trabajo.

2.1. Juegos y videojuegos

Por juego se suele entender una actividad realizada por uno o varios jugadores, siguiendo un conjunto de reglas establecidas en común, con un objetivo lúdico.
Sin adentrarnos demasiado en el debate filosófico de qué es un videojuego y qué no es un videojuego (debate muy vivo en los últimos tiempos en las redes, debido al lanzamiento de diversos títulos que desafían la concepción normal de videojuego, como Gone Home en el que no hay condición de derrota, o Johan Sebastian Joust, que se juega con los mandos del PSMove pero sin pantalla), podemos decir que un videojuego es un juego que se ejecuta en un soporte electrónico. El tipo de videojuegos que nos ocupa tendrá una salida de imagen a través de alguna pantalla, y recibirá órdenes de uno o varios jugadores humanos a través de algún dispositivo hardware dedicado a ello, como puede ser una pantalla táctil, un mando, un teclado o un ratón. En este trabajo nos centraremos en los llamados videojuegos en tiempo real, en los que las acciones de los jugadores no se realizan por turnos, sino que tienen lugar en un instante cualquiera dentro del continuo de la simulación.

2.2. Pathfinding

Llamamos pathfinding a la búsqueda del camino más corto entre dos puntos, aunque en el ámbito de los videojuegos se le da más importancia a la búsqueda de un camino que sortee un obstáculo que al hecho de que dicho camino sea efectivamente el más corto. Esto se debe principalmente a dos motivos: en primer lugar, en videojuegos no se dispone de mucho tiempo de procesamiento para calcular un camino, por lo que se prefiere un algoritmo que proporcione una solución aceptable en un tiempo mínimo frente a un algoritmo que proporcione una solución óptima pero en un tiempo mayor; en segundo lugar, generalmente el camino óptimo no va a resultar tan natural como otros caminos.
Existen multitud de algoritmos de pathfinding, pero para este trabajo nos hemos centrado en los más destacables.

2.2.I. A*

A* es sin duda alguna el referente en cuanto a pathfinding. Utiliza una función de evaluación que combina por una parte el coste desde el nodo origen hasta el nodo actual y por otra una función heurística que estima el coste desde el nodo actual hasta el nodo meta. La clave del éxito de A* es que, si el heurístico utilizado es admisible, es decir, si no sobreestima el coste de alcanzar el nodo meta, el algoritmo garantiza que se encuentra el camino más corto si es que existe, es decir, la solución es óptima.
Además, A* es un algoritmo tremendamente flexible: el coste de computación varía según el heurístico que se utilice. Por ejemplo, si utilizamos un heurístico que siempre dé como resultado 0, el algoritmo se comporta exactamente como el algoritmo de Dijkstra. Si, por otra parte, utilizamos un algoritmo no admisible, podemos obtener una solución (no óptima) en un tiempo mucho menor, como defienden algunos autores[1]. En este trabajo se ha optado por usar un heurístico admisible correspondiente a la distancia en movimiento 8-vecinos entre dos puntos.

2.2.II. HPA*

HPA* es un algoritmo de los que llamamos jerárquico: divide el mapa de juego en bloques, y aplica la búsqueda de caminos en varios niveles, empezando por el superior y utilizando la información de niveles inferiores para calcular caminos locales. [2]

2.2.III. Suavizado

Por lo general, y de forma simplificada, A* se suele implementar sobre una cuadrícula con un movimiento en cuatro u ocho direcciones. En los casos en que el espacio de juego no es discreto sino continuo y los personajes pueden moverse en cualquier dirección, se pueden aplicar postprocesados sobre la solución de A* que simplifican el camino, convirtiendo en líneas rectas secciones que originalmente serían un zigzag, por ejemplo. Existen también algoritmos como el llamado Theta* que incorporan este tipo de movimiento en cualquier dirección, pero en este trabajo se ha optado por aplicar un postprocesado sobre A* por reutilización de código.

2.3. Movimiento

Para el movimiento autónomo de un agente controlado por una inteligencia artificial se usa una combinación de diversas técnicas. En este trabajo se han utilizado steering behaviors combinados con una máquina de estados para la toma de decisiones.

[1] http://realtimecollisiondetection.net/blog/?p=56
[2] http://aigamedev.com/open/review/near-optimal-hierarchical-pathfinding/

2.3.I. Steering behaviors

Los llamados steering behaviors son comportamientos que, agregados, determinan cuál será la posición del agente en base a su posición y velocidad actual, su entorno y sus objetivos. Cada steering behavior cumple una función, y será necesario un nivel superior de decisión que determine qué comportamientos utilizar en cada momento de juego. De forma abstracta, cada comportamiento recibe la información que necesite del agente o del entorno y dará como resultado un vector que representa una fuerza. Al sumarse los distintos vectores de los comportamientos activos, se obtienen comportamientos con una apariencia compleja y un cómputo muy sencillo.

Huida

El comportamiento de huida aplica una fuerza que intenta alejar al agente del punto que se proporcione como entrada. Existe una variante llamada evasión que aplica una predicción sobre la posición futura del elemento que se pretende evitar.

Búsqueda

El comportamiento de búsqueda aplica una fuerza que intenta acercar al personaje al punto que se proporcione. Al igual que en el caso anterior, existe una variante llamada persecución que aplica una predicción sobre la posición futura del elemento que se persigue.

Deambular

El comportamiento de deambular intenta que el agente se desplace de forma aleatoria pero no caótica, evitando los giros bruscos tanto como las líneas perfectamente rectas.

2.3.II. Máquina de estados

En el apartado anterior mencionábamos que es necesario acompañar a los steering behaviors de un nivel superior de decisión que determine qué comportamientos deben estar activos en cada momento y cómo se combinan. Por lo general se utilizan tres aproximaciones: máquinas de estados, árboles de decisión y funciones de utilidad. La más sencilla de estas opciones es la máquina de estados, que se compone de una serie de estados, las transiciones entre estos estados, las condiciones para que se den estas transiciones, y los eventos que dispararán las condiciones.

3 - Diseño de la aplicación

En este capítulo se explica cómo se ha procedido con el diseño de la aplicación que se ha desarrollado. Vamos a hablar del entorno de Unity3D, y luego de cómo se ha enfocado la implementación de los algoritmos de pathfinding, de movimiento y la máquina de estados.

3.1. Visión general

La aplicación se ha separado en dos secciones principales: una centrada en los algoritmos de pathfinding, en la que se pueden hacer comparativas de los distintos algoritmos, y otra centrada en los steering behaviors, donde se realiza un enfrentamiento entre un jugador humano y un jugador controlado por una inteligencia artificial. Un menú inicial permite navegar entre ambas secciones.
El menú principal incluye tres botones: uno lleva a la escena de enfrentamiento, otro lleva a la escena de pathfinding, y el último cierra la aplicación. En este menú se incluye también el título del trabajo y los nombres del alumno y el tutor.
En la sección correspondiente al enfrentamiento, donde se muestran los steering behaviors, incluimos los siguientes elementos:

- Un espacio de juego delimitado y con una serie de obstáculos
- Dos objetos móviles correspondientes a los personajes controlados por el jugador humano y la IA, con capacidad de moverse en tiempo real por el escenario
- Un menú inicial donde se pueden modificar diversos parámetros de los personajes, del control por parte del jugador, y de los valores utilizados por la inteligencia artificial

En esta sección, el usuario podrá controlar a uno de los dos personajes y experimentar con el comportamiento del oponente controlado por la IA. Además, modificando los diversos parámetros del menú podrá experimentar cómo afectan distintos valores al comportamiento de la IA.

En la sección de pathfinding se incluyen los siguientes elementos visuales:

- Un espacio de juego delimitado, con obstáculos y con una cuadrícula
- Un objeto móvil similar a los personajes de la sección anterior que recorrerá los caminos introducidos por el usuario

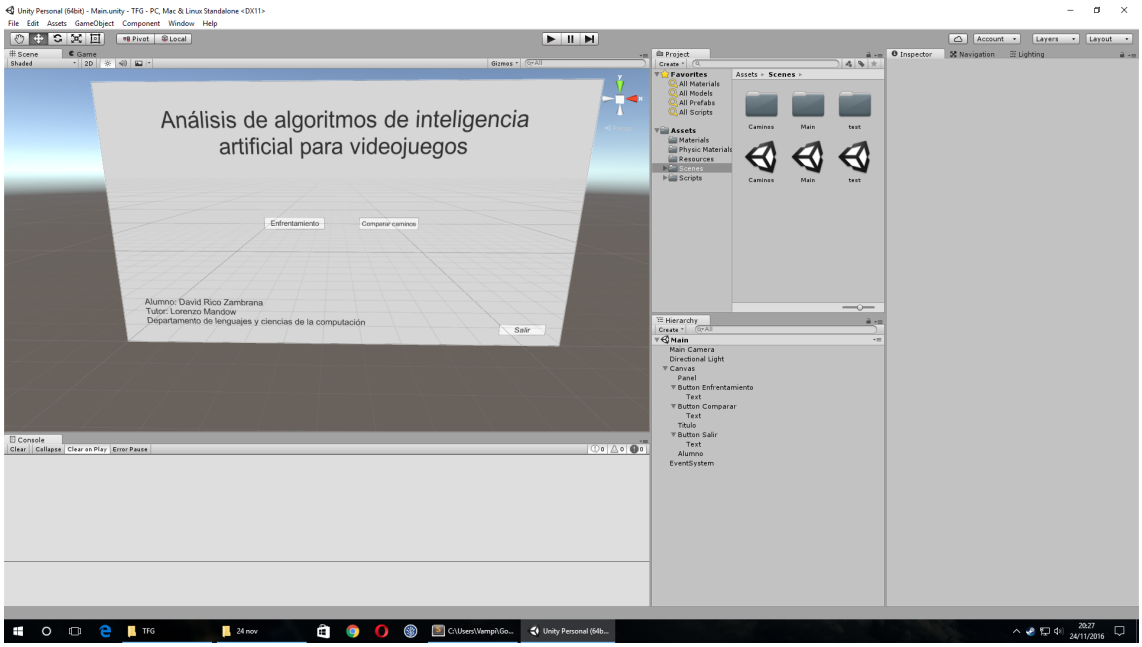

Figura 3.1: Entorno del editor de Unity3D con la escena del menú principal

- Un menú donde se puede seleccionar el algoritmo de pathfinding que se quiere visualizar, y las coordenadas de origen y destino del camino a buscar, y donde se muestran los costes de los caminos encontrados por los distintos algoritmos

En esta sección el usuario podrá visualizar los caminos generados por los distintos algoritmos de pathfinding, además de visualizar al personaje recorriendo el camino correspondiente al algoritmo que se desee; se podrán visualizar además los costes de los distintos caminos y compararlos entre sí.

3.2. Unity3D

En el editor de Unity3D, el contenedor básico en torno al que gira el desarrollo es la escena. Una escena contiene una jerarquía con distintos objetos de juego, entre los que se incluye una cámara, alguna luz, y los elementos que queramos incluir. El menú inicial de la aplicación está hecho en su propia escena, y cada uno de los botones carga la escena correspondiente. Así, el desarrollo de las distintas secciones de la aplicación es independiente de las demás.

En la figura 3.1 se muestra el entorno del editor de Unity3D. Se ha cargado la escena correspondiente al menú inicial, que se puede ver en la ventana de la escena. En la parte de la derecha de la pantalla se encuentran las vistas del proyecto, de la jerarquía de la escena, y del inspector de objetos. En la jerarquía se puede ver que la escena consiste en una luz, una cámara, y un elemento Canvas que contiene todos los objetos correspondientes al menú. En Unity3D, los elementos de interfaz de usuario (botones, etiquetas, etc) van todos dentro de algún Canvas; una misma escena puede tener varios Canvas, lo cual

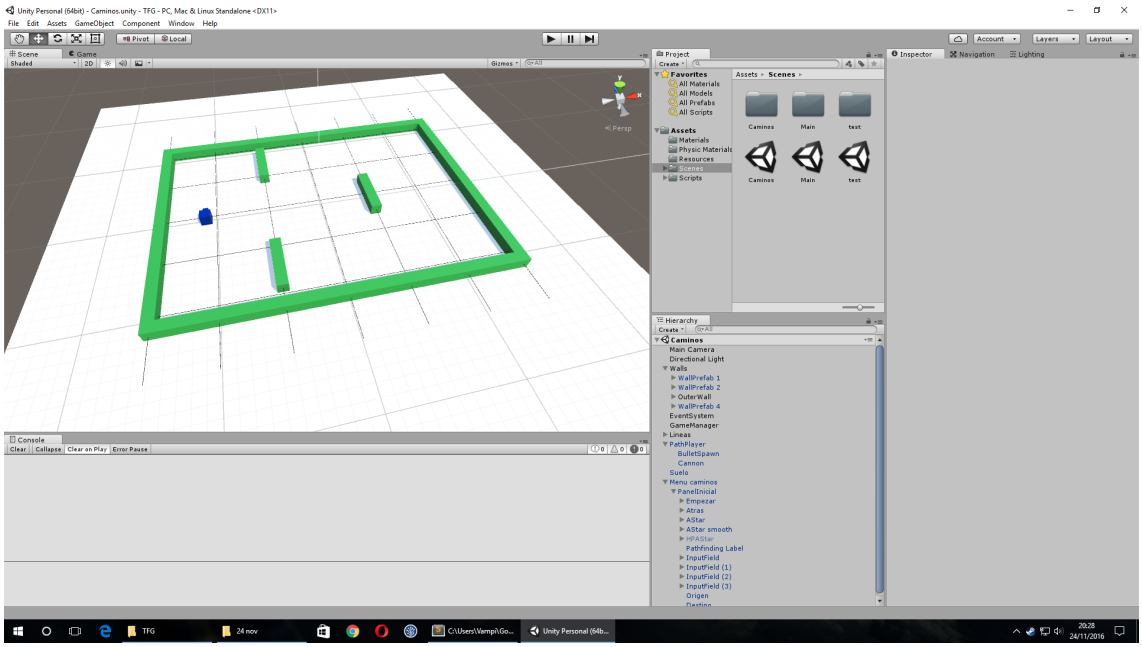

Figura 3.2: Entorno del editor de Unity3D con la escena de la comparativa de caminos

puede ser útil para mostrar u ocultar diferentes menús o partes del interfaz. En este caso, el Canvas es sencillo: hay tres botones y unas cuantas etiquetas de texto.

En la figura 3.2 se muestra la escena del comparador de caminos. En la vista de escena se puede ver el escenario, con el personaje, los obstáculos, etc. En la vista de la jerarquía se puede ver que también hay un Canvas con los elementos del menú, aunque en esta ocasión no se ve en la vista de escena. Se puede apreciar que este menú es más complejo que el anterior: hay varios inputfields, además de los botones y etiquetas de texto. En esta captura podemos ver además que el entorno de trabajo de Unity3D es bastante gráfico: los elementos de juego se introducen en la escena de forma visual y prácticamente se montan tal y como se van a ver durante la ejecución.

En la figura 3.3 vemos la escena correspondiente al enfrentamiento entre el jugador humano (el personaje azul en la escena) y el jugador controlado por la inteligencia artificial (el personaje rojo en la escena). Vemos que la escena en sí no se diferencia mucho de la anterior, y la diferencia principal es el comportamiento de los personajes, que en la anterior escena estaba diseñado para seguir un camino en función del algoritmo de entrada que se le proporcione, y en esta escena están diseñados para el enfrentamiento.

3.3. Pathfinding

Para los algoritmos de pathfinding se ha optado por una arquitectura con una superclase Pathfinding, que incluye un método GetPath que, tras introducirle las coordenadas de origen y destino devuelve un camino entre ambos puntos, y otro método DrawPath que

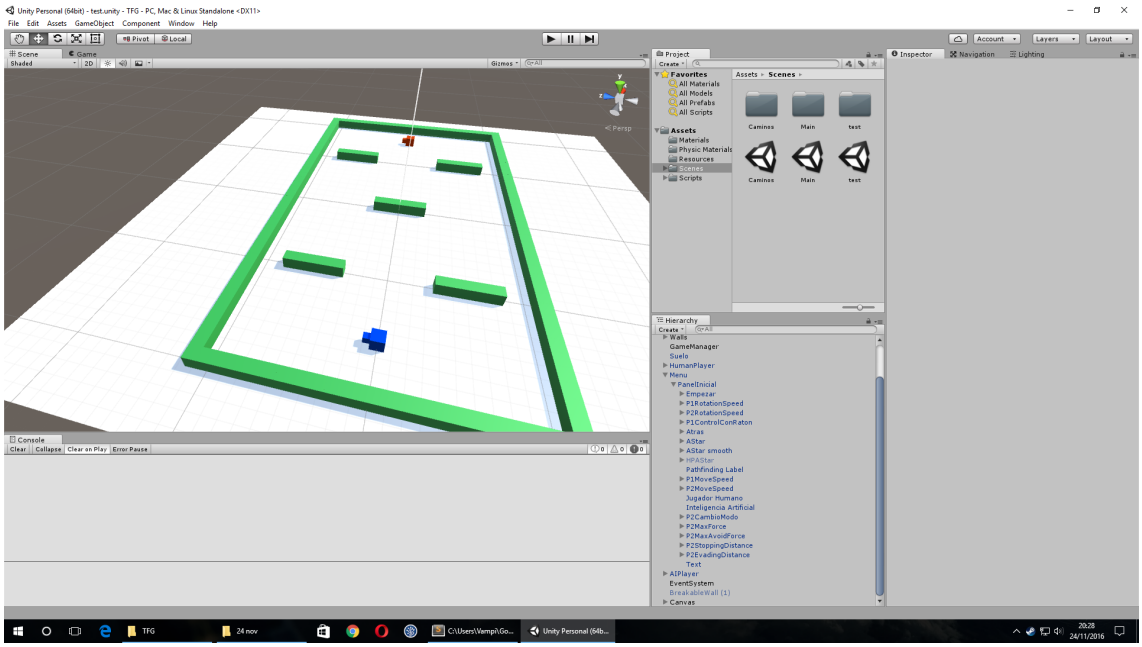

Figura 3.3: Entorno del editor de Unity3D con la escena del enfrentamiento

hace lo mismo pero dibujando el camino en pantalla. Se ha creado una clase TerrainInfo cuya responsabilidad es tomar la información del escenario de juego y convertirla en una matriz de celdas que los algoritmos de pathfinding puedan entender y trabajar.

3.4. Movimiento

Los comportamientos de movimiento se han implementado todos en una clase SteeringBehaviorManager; la inteligencia artificial del jugador rojo, implementada en una clase AIPlayerBasic, hace llamadas a este SteeringBehaviorManager para acumular los vectores de los comportamientos que necesite, según dicte la máquina de estados.

3.5. Máquinas de estados

La máquina de estados está programada en la misma clase AIPlayerBasic. En función de determinados valores, como por ejemplo el número de veces que el personaje haya recibido un impacto, la máquina puede saltar de un estado a otro. Actualmente la máquina tiene tres estados: uno agresivo, en el que los comportamientos tienen como objetivo acercar al personaje a su oponente; uno evasivo, en el que los comportamientos intentan alejar al personaje de su oponente; y un estado entre búsqueda y wandering, que se activa al perder línea de visión hacia su oponente, y que consiste en separar el escenario en sectores y mover al personaje de un sector a uno de los sectores adyacentes aleatoriamente.

4 - Uso de la aplicación

En la figura 4.1 se muestra el menú inicial de la aplicación. En el centro de la pantalla hay dos botones: Enfrentamiento y Comparar caminos. El primero, Enfrentamiento, nos lleva a la sección de la aplicación donde se muestra el funcionamiento de los algoritmos de movimiento y la máquina de estados. El segundo, Comparar caminos, nos lleva a la sección en la que se compara la ejecución de los distintos algoritmos de pathfinding implementados. Por último, en la esquina inferior derecha encontramos el botón Salir, con el que cerramos la aplicación.

Figura 4.1: Menú inicial

Figura 4.2: Menú de enfrentamiento

4.1. Experimentación con movimiento y máquinas de estado

La figura 4.2 muestra el menú de la sección de enfrentamiento, en la que un jugador humano se enfrenta a un jugador controlado por una inteligencia artificial. A la izquierda tenemos los controles de los parámetros del jugador humano, y a la derecha los controles de los parámetros de la inteligencia artificial.
Empezando por la izquierda, con el jugador humano, tenemos un slider que controla la velocidad de rotación del personaje, y otro slider que controla la velocidad a la que se puede desplazar el personaje. A continuación, una casilla nos permite indicar si queremos controlar al personaje con el ratón o, si no la marcamos, con el teclado.
Si elegimos el control con ratón, haciendo click izquiero sobre el escenario ordenamos al personaje que se mueva a esa posición, y haciendo click derecho dispararemos en la dirección del puntero. Si elegimos este modo de control, tendremos que seleccionar además el algoritmo de pathfinding que va a usar el personaje para desplazarse de su posición a la que le indiquemos. Por otra parte, si preferimos el control con teclado, usaremos las flechas para desplazar al personaje, las teclas W A S D para mirar hacia arriba, izquierda, abajo y derecha respectivamente, y la barra de espacio para disparar en la dirección en que estemos mirando.
A la derecha tenemos los parámetros de la inteligencia artificial. En primer lugar, los mismos sliders que para el jugador humano controlan la velocidad de rotación y de desplazamiento del personaje. Un siguiente slider controla cada cuántos impactos recibidos va a cambiar el comportamiento de la inteligencia artificial, entre un modo agresivo y un

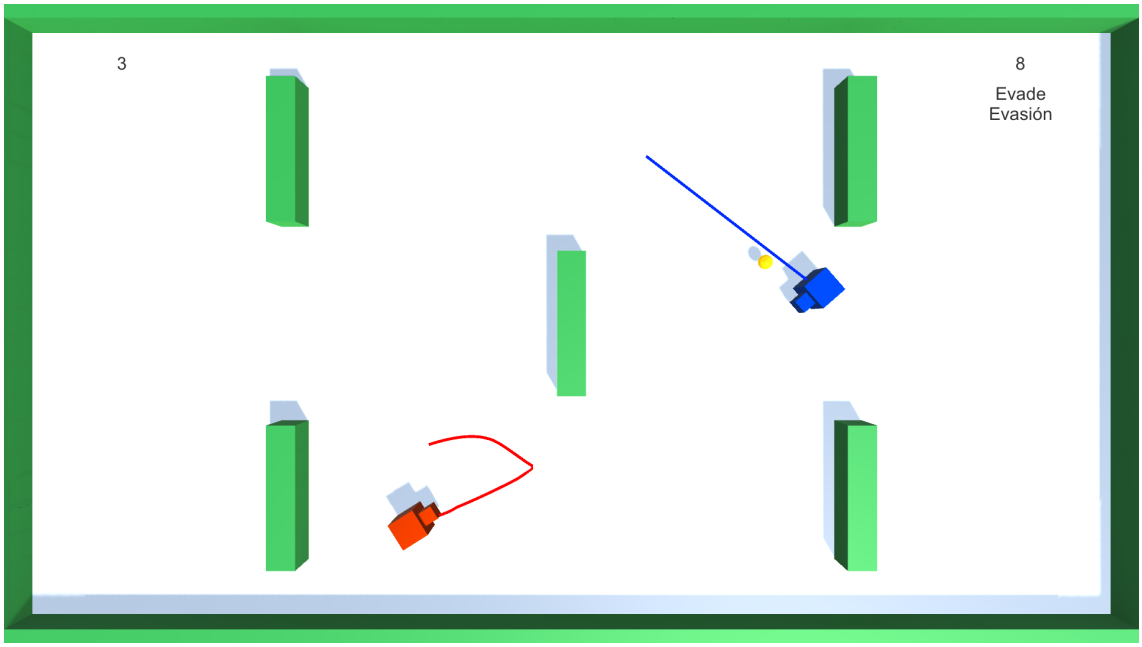

Figura 4.3: Escenario de enfrentamiento

modo evasivo. Por último, dos sliders controlan el tamaño del vector fuerza resultado de agregar los distintos steering behaviors, y el tamaño del vector fuerza del comportamiento de evasión de obstáculos, y otros dos sliders controlan la distancia hasta la que intenta acercarse el personaje en modo agresivo y a la que intenta alejarse en el modo evasivo.

En la figura 4.3 vemos la pantalla de juego en sí. El personaje azul es el controlado por el jugador humano, y el personaje rojo corresponde a la inteligencia artificial. Arriba, a la izquierda aparecen los impactos que ha logrado el jugador humano, y a la derecha los impactos que ha logrado la inteligencia artificial, así como el estado actual de la máquina de estados que lo controla.

4.2. Experimentación con pathfinding

La figura 4.4 muestra la sección de pathfinding, donde se comparan los distintos algoritmos de búsqueda de caminos. A la derecha aparece el menú que controla esta sección. Mediante una casilla podemos seleccionar qué algoritmo queremos ver ejecutar al personaje, aunque mediante una traza se muestran todos los caminos de los distintos algoritmos a la vez. Podemos seleccionar las coordenadas de origen y destino del camino de dos formas: bien introduciendo los números en los recuadros que aparecen, o bien haciendo click izquierdo sobre el escenario para seleccionar el origen, y click derecho para seleccionar el destino.

En la figura 4.5 vemos el resultado tras elegir un camino y pulsar el botón Empezar. Los números que aparecen junto a cada casilla de los distintos algoritmos corresponde al coste del camino encontrado. Sobre el escenario vemos las líneas de colores que corresponden a cada camino.

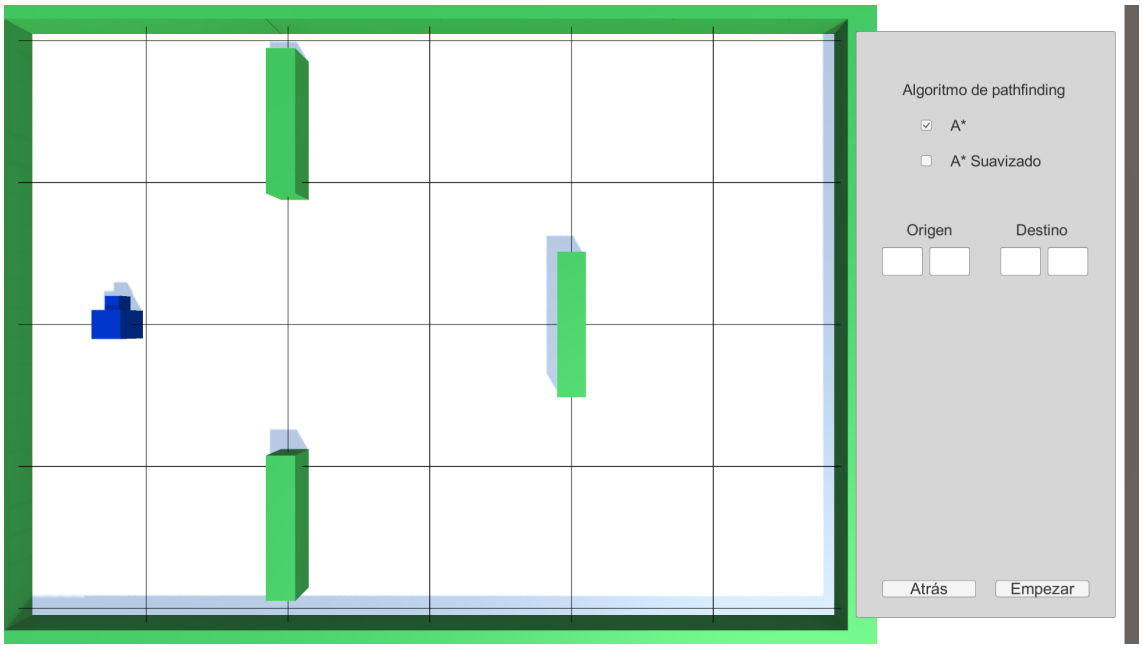

Figura 4.4: Pantalla de pathfinding

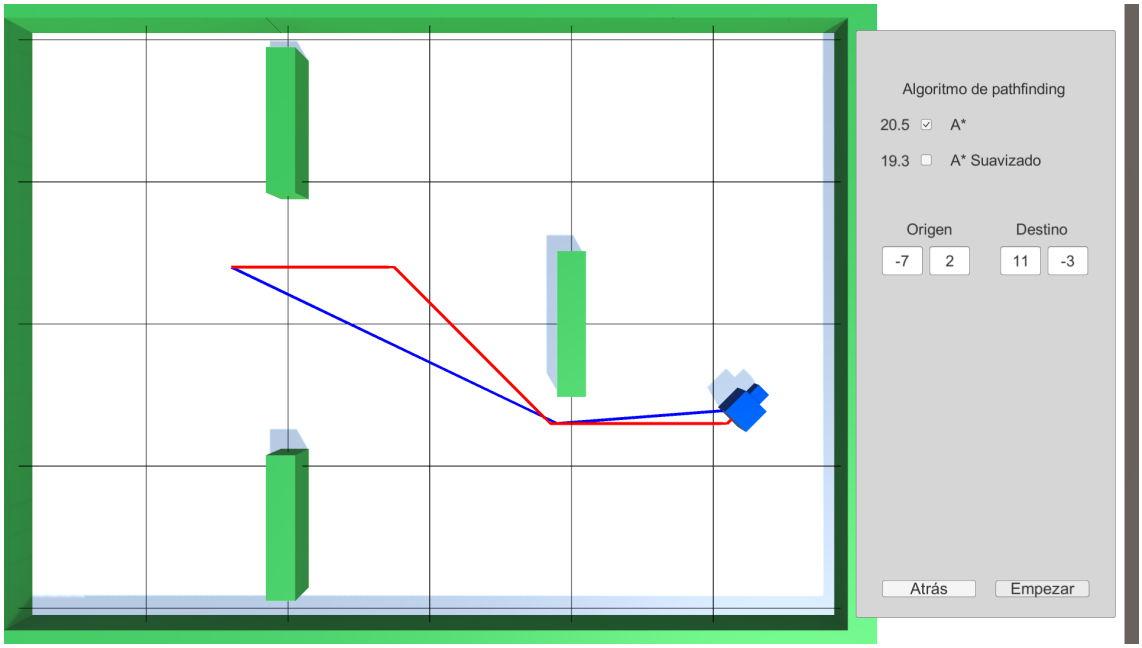

Figura 4.5: Pantalla de pathfinding tras ejecución de un camino

5 - Conclusiones y trabajo futuro

A modo de conclusión, podemos considerar que los objetivos principales de este trabajo se han cumplido correctamente: se ha desarrollado una aplicación en Unity3D, y en las distintas secciones de la aplicación se pueden visualizar de forma didáctica los resultados de los diversos algoritmos de búsqueda de caminos y los comportamientos y cambios en un agente controlado por steering behaviors.
El desarrollo de este trabajo ha supuesto un acercamiento a Unity3D, uno de los entornos de trabajo más importante en la industria del videojuego actualmente, y a la programación en C#. Unity3D ha demostrado ser una herramienta flexible y potente, con una curva de aprendizaje suave, que permite obtener resultados vistosos con relativamente poco trabajo.
En cuanto a los algoritmos de búsqueda de caminos, la comparativa entre los distintos caminos resulta especialmente ilustrativa. Se puede ver que el algoritmo A*, a pesar de proporcionar caminos óptimos dentro de sus condiciones, está limitado de forma natural por las direcciones de movimiento que se le proporcionen. En un entorno con un espacio continuo, como el que se presenta en este trabajo, da resultados mucho más naturales un algoritmo con un suavizado, que permita movimientos en cualquier ángulo.
Por último, los steering behaviors han demostrado ser muy delicados en cuanto a los valores de los parámetros que se utilizan para determinar las fuerzas de los distintos vectores. Esto se debe, entre otras cosas, a que el fuerte de los steering behaviors son los comportamientos en grupo, mientras que para este trabajo se han utilizado de forma individual. Cuando se aplican a un número suficientemente grande de agentes, la masa se comporta de forma suficientemente convincente, aunque individualmente haya errores o comportamientos extraños.
A partir de este trabajo, sería interesante seguir incluyendo algoritmos de búsqueda de caminos para ampliar la comparativa. Por otra parte, sería interesante ampliar la sección de steering behaviors para incluir varios agentes que trabajen en grupo, y poder así comprobar los comportamientos grupales.

6 - Apéndice: ampliaciones de la aplicación

La aplicación es fácilmente ampliable en varias direcciones. En primer lugar, se pueden añadir distintas escenas en las que aplicar otros conceptos que no se han tratado en este trabajo. Por ejemplo, se podría añadir una escena en la que hacer una demostración de aprendizaje automático. Para ello bastaría con crear la nueva escena en Unity3D, y en la escena del menú inicial añadir un botón que cargue esa escena.
En segundo lugar, se pueden ampliar las escenas existentes: es muy fácil añadir más algoritmos de pathfinding, por ejemplo: ya tenemos las clases para los nodos, la estructura para gestionar los distintos algoritmos, etc. La clase correspondiente al algorimo que se quiera añadir solo tiene que heredar de Pathfinding e implementar los métodos FindPath y DrawPath.
En cuanto a los steering behaviors, se pueden añadir más comportamientos en la clase manager, y es fácil ampliar la máquina de estados para incluir los nuevos steering behaviors. Habría que programar la lógica de las transiciones, que está en la clase AIPlayerBasic, y enlazar sus valores al menú de juego resulta trivial a través del editor de Unity3D.

7 - Bibliografía

Para el desarrollo de este trabajo se ha recurrido a la siguiente bibliografía:

- Millington, I. *Artificial Intelligence for Games*. San Francisco: Morgan Kaufmann Publishers Inc., 2006. ISBN 0124977820.
- Russell, S. y Norvig, P. *Artificial Intelligence: a modern approach (3rd Ed.)*. New Jersey: Pearson Education Inc., 2010. ISBN 0132071487.
- Murray, J.W. *C# Game Programming Cookbook for Unity 3D*.CRC Press, 2015. ISBN 1466581409.
- Web: http://aigamedev.com/open/review/near-optimal-hierarchical-pathfinding/ Fecha de consulta: noviembre 2016.
- Web: http://theory.stanford.edu/ãmitp/GameProgramming/Variations.html Fecha de consulta: noviembre 2016.
- Web: https://www.hindawi.com/journals/ijcgt/2008/873913/ Fecha de consulta: noviembre 2016.
- Web: http://realtimecollisiondetection.net/blog/?p=56 Fecha de consulta: noviembre 2016.
- Web: http://www.cs.ru.nl/bachelorscripties/2013/Linus_van_Elswijk___0710261___Hierarchical_Path-Finding_Theta_star_Combining_HPA_star_and_Theta_star.pdf Fecha de consulta: noviembre 2016.
- Web: http://xfleury.github.io/graphsearch.html Fecha de consulta: noviembre 2016.
- Web: https://gamedevelopment.tutsplus.com/tutorials/understanding-steering-behaviors-movement-manager–gamedev-4278 Fecha de consulta: noviembre 2016.
- Web: http://cstheory.stackexchange.com/questions/11855/how-do-the-state-of-the-art-pathfinding-algorithms-for-changing-graphs-d-d-l Fecha de consulta: noviembre 2016.
- Web: http://www.red3d.com/cwr/steer/ Fecha de consulta: noviembre 2016.
- Web: https://davidtheory.wordpress.com/2012/11/04/ia-en-unity3d-steering-behaviours/ Fecha de consulta: noviembre 2016.

Agradecimientos

Este trabajo está especialmente dedicado a aquellas personas que me han empujado cuando más lo necesitaba, que me han soportado cuando no lo merecía, y que me han ayudado a superar una piedra tras otra.
Por supuesto, a mis padres, que no solo me apoyan moralmente sino que además ponen el dinero, y son un ejemplo y una referencia en mi vida.
A mis hermanos, Ale y Alberto, que entienden mejor que nadie la presión de la recta final, las noches sin dormir, y el impulso de tirarlo todo y volver a empezar cuando ya no queda tiempo. Y a Jesús, que junto a mis hermanos forman mi equipo de trabajo, y que están llevando nuestros proyectos adelante mientras yo termino este trabajo.
A Marina. Tú me has dado el último empujón, el más importante, cuando ya había decidido abandonar. Me has animado, has estado más pendiente que una madre y has confiado en mí más que yo mismo. No hay café suficiente para pagártelo. Y a Gema y María también, porque entre las tres me podéis alegrar hasta el peor de los días. No cambiéis.
Y a Lorenzo, que ha sido un magnífico profesor y, probablemente, el mejor tutor que podría haber tenido para este trabajo.
Gracias a todos.